WHAT IS CLAIMED IS:

1. A method for providing business continuity comprising:
detecting that a first application of a plurality of applications running on a first system of a plurality of systems within a cluster has failed;
determining whether the first application can be failed over to a second system of the systems;
when the first application can be failed over,
restarting the first application on the second system; and
when the first application cannot be failed over,
determining whether a third system of the systems satisfies a first prerequisite for the first application; and
when the third system satisfies the first prerequisite,
moving the first application to the third system.

2. The method of claim 1 wherein
the determining whether the first application can be failed over further comprises:
determining whether the first application will overload the second system; and
when the first application will overload the second system,
determining that the first application cannot be failed over.

3. The method of claim 1 wherein
the determining whether the first application can be failed over further comprises:
determining whether the first application will overload the second system; and
when the first application will overload the second system,
providing an overload warning.

4. The method of claim 1 wherein
the determining whether the third system satisfies the first prerequisite further comprises:
determining whether the first application will overload the third system; and
when the first application will overload the third system,
determining that the third system does not satisfy the first prerequisite.

5. The method of claim 1 wherein

the third system satisfies the first prerequisite when the third system has an available resource fulfilling the first prerequisite.

6. The method of claim 1 wherein
the first prerequisite corresponds to a load placed on the third system by running the first application; and
the third system satisfies the first prerequisite when the available resource corresponds to an available capacity greater than or equal to the load.

7. The method of claim 1 further comprising:
ascertaining that the first application falls within a limit for the third system prior to the moving.

8. The method of claim 7 wherein
the limit corresponds to a maximum number of the applications that can be run on the third system.

9. The method of claim 7 wherein
the limit corresponds to a maximum number of the applications of a type of the first application that can be run on the third system.

10. The method of claim 1 further comprising:
when the third system does not satisfy the first prerequisite,
    determining whether a second application of the applications running on the third system of the systems can be moved to free a resource for running the first application;
    when the second application can be moved,
        determining whether a fourth system of the systems satisfies a second prerequisite for the second application of the applications; and
        when the fourth system satisfies the second prerequisite,
            moving the second application to the fourth system; and
            moving the first application to the third system.

11. The method of claim 1 further comprising:
determining whether a second application of the applications can be moved;
when the second application can be moved,

determining whether a fourth system of the systems satisfies a second prerequisite for the second application of the applications; and

when the fourth system satisfies the second prerequisite,

moving the second application to the fourth system.

12. The method of claim 1 further comprising:

ascertaining that the second application falls within a limit of the fourth system prior to the moving the second application.

13. The method of claim 1 further comprising:

using a first priority of the first application among the applications and a second priority of the second application among the applications for determining whether the second application can be moved.

14. The method of claim 1 wherein

detecting whether the first application has failed further comprises at least q0one of:

simulating that the first application has failed; and

simulating that the first system has failed.

15. A method for providing business continuity comprising:

detecting that a first application of a plurality of applications is to be started;

determining whether a first system of a plurality of systems in a cluster meets a first prerequisite for the first application;

when the first system meets the first prerequisite,

starting the first application on the first system; and

when the first system does not meet the first prerequisite,

determining whether a second system of the systems satisfies the first prerequisite; and

when the second system satisfies the first prerequisite,

starting the first application on the second system.

16. The method of claim 15 wherein

the detecting that the first application is to be started comprises detecting that the first application has failed while running on a third system of the systems.

17. The method of claim 15 wherein

the detecting that the first application is to be started comprises detecting that the first application is running on a third system of the systems, the third system being overloaded.

18. A system comprising:

means for detecting that a first application of a plurality of applications running on a first system of a plurality of systems within a cluster has failed;

first determining means for determining whether the first application can be failed over to a second system of the systems;

means for restarting the first application on the second system;

second determining means for determining whether a third system of the systems satisfies a first prerequisite for the first application; and

means for moving the first application to the third system.

19. The system of claim 18 further comprising:

third determining means for determining whether a second application of the applications can be moved;

fourth determining means for determining whether a fourth system of the systems satisfies a second prerequisite for the second application of the applications; and

second moving means for moving the second application to the fourth system.

20. The system of claim 19 further comprising:

means for using a first priority of the first application among the applications and a second priority of the second application among the applications for determining whether the second application can be moved.

21. The system of claim 18 wherein

the means for detecting whether the first application has failed further comprises:

means for simulating that the first application has failed; and

means for simulating that the first system has failed.

22. A computer program product comprising:

detecting instructions to detect that a first application of a plurality of applications running on a first system of a plurality of systems within a cluster has failed;

first determining instructions to determine whether the first application can be failed over to a second system of the systems;

restarting instructions to restart the first application on the second system;
second determining instructions to determine whether a third system of the systems satisfies a first prerequisite for the first application;
moving instructions to move the first application to the third system; and
a computer-readable medium for storing the detecting instructions, the first determining instructions, the restarting instructions, the second determining instructions, and the moving instructions.

23. The computer program product of claim 21 further comprising:
third determining instructions to determine whether a second application of the applications can be moved;
fourth determining instructions to determine whether a fourth system of the systems satisfies a second prerequisite for the second application of the applications; and
second moving instructions to move the second application to the fourth system;
wherein
the computer-readable medium further stores the third determining instructions, the fourth determining instructions, and the second moving instructions.

24. The computer program product of claim 23 further comprising:
using instructions to use a first priority of the first application among the applications and a second priority of the second application among the applications to determine whether the second application can be moved;
wherein
the computer-readable medium further stores the using instructions.

APPENDIX A

Examples

The following example uses Limits and Prerequisites to control the total number of application groups that may run on a system. The cluster consists of four similar servers. There are five application groups, which are roughly equivalent in requirements for processing power and in the amount of Load each application group requires of a system. Each server can host two such application groups. This example does not use application group Load and system Capacity. Also, the application groups use a default AutoStartPolicy and FailOverPolicy.

Example Configuration File with Limits

```
system Svr1 (
Limits = {GroupWeight = 2}
)

system Svr2 (
Limits = {GroupWeight = 2}
)

system Svr3 (
Limits = {GroupWeight = 2}
)

system Svr4 (
Limits = {GroupWeight = 2}
)

group G1 (
SystemList = { Svr1, Svr2, Svr3, Srv4}
AutoStartList = { Svr1, Svr2 }
Prerequisites = { GroupWeight = 1}
)

group G2 (
SystemList = { Svr1, Svr2, Svr3, Svr4}
AutoStartList = { Svr2, Svr3 }
Prerequisites = { GroupWeight = 1 }
)

group G3 (
```

```
SystemList = { Svr1, Svr2, Svr3, Svr4}
AutoStartList = {Svr3, Svr4 }
Prerequisites = { GroupWeight = 1 }
)

group G4 (
SystemList = { Svr1, Svr2, Svr3, Svr4}
AutoStartList = { Svr4, Svr1 }
Prerequisites = { GroupWeight = 1 }
)

group G5 (
SystemList = { Svr1, Svr2, Svr3, Svr4}
AutoStartList = { Svr2, Svr3 }
Prerequisites = { GroupWeight = 1 }
)
```

AutoStart Operation

This example uses the default AutoStartPolicy = Order. Application groups are brought online on the first system available in the AutoStartList. In this way, G1 will start on Svr1, G2 on Svr2, and so on. G5 will start on Svr2.

Normal Operation

An example cluster configuration (assuming all systems are running) is provided below:

Svr1

CurrentLimits = {GroupWeight=1}
(Group G1)

Svr2

CurrentLimits = {GroupWeight=0}
(Groups G2 and G5)

Svr3

CurrentLimits = {GroupWeight=1}
(Group G3)

Svr4

CurrentLimits = {GroupWeight=1}
(Group G4)

Failure Scenario

In the first failure scenario, assume Svr2 fails. With application groups G2 and G5 configured with an identical SystemList, both application groups can run on any system. The cluster management application can serialize the choice of failover nodes for the two groups. G2, being canonically first, is started on Svr1, the lowest priority in the SystemList, thereby exhausting the Limits for Svr1. G5 is then started on the next system in the order of the SystemList for group G5. G5 goes online on Svr3. Following the first failure, the cluster now looks like the following:

```
Svr1

        CurrentLimits = {GroupWeight=0 }
        (Groups G1 and G2)

Svr3

        CurrentLimits = {GroupWeight=0}
        (Groups G3 and G5)

Svr4

        CurrentLimits = {GroupWeight=1}
        (Group G4)
```

Cascading Failures

Assuming Svr2 cannot immediately repaired, the cluster can tolerate the failure of an individual application group on Svr1 or Svr3, but no further node failures.

Load-Based example

The following sample cluster shows the use of simple load based startup and failover. SystemZones, Limits and Prerequisites are not used.

The cluster consists of four identical systems, each with the same capacity. Eight application groups, G1-G8, with various loads run in the cluster.

Configuration File

```
        include "types.cf"
        cluster SGWM-demo

system Svr1 (
        Capacity = 100
        )
```

```
system Svr2 (
        Capacity = 100
        )

system Svr3 (
        Capacity = 100
        )

system Svr4 (
        Capacity = 100
        )

group G1 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 20
        )

group G2 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 40
                )

group G3 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 30
                )

group G4 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 10
                )
```

```
group G5 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 50
                )

group G6 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 30
                )

group G7 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 20
                )

group G8 (

        SystemList = { Svr1, Svr2, Svr4, Svr4 }
        AutoStartPolicy = Load
        AutoStartList = { Svr1, Svr2, Svr3, Svr4 }
        FailOverPolicy = Load
        Load = 40
                )
```

AutoStart Operation

As mentioned above, application groups can be placed in a queue as soon as they are started on a system. For the purposes of this example, application groups are placed into the queue in the same order that the application groups are described, G1 through G8.

G1 is started on the system with the highest AvailableCapacity. Since the systems are equal, Svr1 is chosen since it is canonically first. G2-G4 start on Svr2 through Svr4. At this time, with the first 4 group startup decisions made, the cluster looks as follows:

Svr1
AvailableCapacity=80

Svr2
AvailableCapacity=60

Svr3
AvailableCapacity=70

Svr4
AvailableCapacity=90

As the remaining application groups are brought online, G5 starts on Svr4, as it has the highest AvailableCapacity. G6 are brought starts on Svr1, with 80 remaining. G7 starts on Svr3, with AvailableCapacity=70. G8 starts on Svr2, with AvailableCapacity=60.

Normal Operation

The final cluster configuration (assuming the original queue of G1-G8) is shown below:

Svr1

AvailableCapacity=50
(Groups G1 and G6)

Svr2

AvailableCapacity=20
(Groups G2 and G8)

Svr3

AvailableCapacity=50
(Groups G3 and G7)

Svr4

AvailableCapacity=40
(Groups G4 and G5)

In this configuration, an overload warning is provided for Svr2 after the default 900 seconds since Svr2 has a default LoadWarningLevel of 80%.

Failure Scenario

In the first failure scenario, assume Svr4 fails, immediately queuing G4 and G5 for failure decision. G4 starts on Svr1, as Svr1 and Svr3 have AvailableCapacity=50 and Svr1 is canonically first. G5 goes online on Svr3. Svr1 Failure decisions are made serially, actual online and offline operations are not. Serializing the failover choice allows complete load-based control, and, in one embodiment, adds less than one second to total failover time.

Following the first failure, the cluster configuration is shown below:

Svr1

AvailableCapacity=40
(Groups G1, G6 and G4)

Svr2

AvailableCapacity=20
(Groups G2 and G8)

Svr3

AvailableCapacity=0
(Groups G3, G7 and G5)

In this configuration, an overload warning is provided for Svr3 to notify an operator or administrator that Svr3 is overloaded. The operator can switch G7 to Svr1 to balance loading across G1 and G3. As soon as Svr4 is repaired, Svr4 rejoins the cluster with an AvailableCapacity=100. Svr4 can then server as a failover target for further failures.

Cascading Failures

Assuming Svr4 is not immediately repaired, further failures are possible. For this example, assume Svr3 now fails. Each application group G3, G5 and G7 is re-started on respective server Svr1, Svr2, and Svr1 These re-starts result in the following configuration:

Svr1
AvailableCapacity= -10
(Groups G1, G6, G4, G3 and G7)

Svr2
AvailableCapacity= -30
(Groups G2 and G8 and G5)

This example shows that AvailableCapacity is a soft limit, and can fall below zero.

Complex 4-system Example

The following example shows a 4-system cluster using multiple system Capacities and various Limits. The cluster consists of two large Enterprise servers (LgSvr1 and LgSvr2) and two Medium servers (MedSvr1 and MedSvr2). Four application groups, G1 through G4, are provided with various Loads and Prerequisites. G1 and G2 are database application groups, with specific shared memory and semaphore requirements. G3 and G4 are middle-tier application groups with no specific memory or semaphore requirements and simply add load to a given system.

Example Configuration File

```
include "types.cf"
cluster Demo (
        )

system LgSvr1 (

        Capacity = 200
        Limits = { ShrMemSeg=20, Semaphores=100, Processors=12}
        LoadWarningLevel = 90
        LoadTimeThreshold = 600
        )
```

```
system LgSvr2 (

        Capacity = 200
        Limits = { ShrMemSeg=20, Semaphores=100, Processors=12 }
        LoadWarningLevel=70
        LoadTimeThreshold=300
        )

system MedSvr1 (

        Capacity = 100
        Limits = { ShrMemSeg=10, Semaphores=50, Processors=6}
        )

system MedSvr2 (

        Capacity = 100
        Limits = { ShrMemSeg=10, Semaphores=50, Processors=6 }
        )

group G1 (

        SystemList = { LgSvr1, LgSvr2, MedSvr1, MedSvr2 }
        SystemZones = { LgSvr1=0, LgSvr2=0, MedSvr1=1, MedSvr2=1 }
        AutoStartPolicy = Load
        AutoStartList = { LgSvr1, LgSvr2 }
        FailOverPolicy = Load
        Load = 100
        Prerequisites = { ShrMemSeg=10, Semaphores=50, Processors=6 }

                )

group G2 (

        SystemList = { LgSvr1, LgSvr2, MedSvr1, MedSvr2 }
        SystemZones = { LgSvr1=0, LgSvr2=0, MedSvr1=1, MedSvr2=1 }
        AutoStartPolicy = Load
        AutoStartList = { LgSvr1, LgSvr2 }
        FailOverPolicy = Load
        Load = 100
        Prerequisites = { ShrMemSeg=10, Semaphores=50, Processors=6 }

)

group G3 (

        SystemList = { LgSvr1, LgSvr2, MedSvr1, MedSvr2 }
        SystemZones = { LgSvr1=0, LgSvr2=0, MedSvr1=1, MedSvr2=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2 }
        FailOverPolicy = Load
        Load = 30
```

```
        )

    group G4 (

            SystemList = { LgSvr1, LgSvr2, MedSvr1, MedSvr2 }
            SystemZones = { LgSvr1=0, LgSvr2=0, MedSvr1=1, MedSvr2=1 }
            AutoStartPolicy = Load
            AutoStartList = { MedSvr1, MedSvr2 }
            FailOverPolicy = Load
            Load = 20
                )
```

AutoStart Operation

Using the main.cf example above, the following is one possible outcome of the AutoStart operation:

G1 – LgSvr1
G2 – LgSvr2
G3 – MedSvr1
G4 – MedSvr2

All application groups are assigned to a system when the cluster starts. Application groups G1 and G2 have an AutoStartList of LgSvr1 and LgSvr2. G1 and G2 are queued to go online on one of these servers, based on highest AvailableCapacity. Assuming G1 starts first, G1 is started on LgSvr1 because LgSvr1 and LgSvr2 both have an initial AvailableCapacity of 200, and LgSvr1 is lexically first.

Application groups G3 and G4 are started, respectively, on MedSvr1 and MedSvr2.

Normal Operation

After starting application groups G1 through G4, the resulting configuration is shown below:

```
LgSvr1

        AvailableCapacity=100
        CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6 }

LgSvr2

        AvailableCapacity=100
        CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6}

MedSvr1

        AvailableCapacity=70
```

CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6 }

MedSvr2

AvailableCapacity=80
CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6}

Failure Scenario

For the first failure example, assume system LgSvr2 fails. The cluster management application scans available systems in G2's SystemList having the same SystemZones grouping as LgSvr2. The cluster management application then creates a subset of systems meeting the application group's Prerequisites. In this case, LgSvr1 meets all necessary Limits. G2 is brought online on LgSvr1, resulting in the following configuration:

LgSvr1

AvailableCapacity=0
CurrentLimits = { ShrMemSeg=0, Semaphores=0, Processors=0 }

MedSvr1

AvailableCapacity=70
CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6 }

MedSvr2

AvailableCapacity=80
CurrentLimits = { ShrMemSeg=10, Semaphores=50, Processors=6}

After 10 minutes, (LoadTimeThreshold = 600) the overload warning on LgSvr1 is provided because LoadWarningLevel exceeds 90%.

Cascading Failure Scenario

In this scenario, a further failure of a system can be tolerated, as each system has remaining Limits sufficient to accommodate the application group running on the peer system.

For example, if a failure were to occur with either MedSvr1 or MedSvr2, the other system would be selected as a failover target, as application groups running on the failed system have MedSvr1 and MedSvr2 in their respective SystemZones.

If a failure instead occurred with LgSvr1, with LgSvr2 still offline, the failover of the application groups G1 and G2 are serialized for the failover decision process. In this case, no

systems exist in the database zone. The first group canonically, G1, will be started on MedSvr2, as MedSvr2 meets all Limits and has the highest AvailableCapacity. Group G2 will be started on MedSvr1, as MedSvr1 is the only remaining system meeting the Limits.

Server Consolidation Example

The following example shows a complex 8-node cluster running multiple applications and several large databases. The database servers are all large enterprise systems, LgSvr1, LgSvr2 and LgSvr3. The middle-tier servers running multiple applications are MedSvr1, MedSvr2, MedSvr3, MedSvr4 and MedSvr5.

Example Configuration File

```
include "types.cf"
cluster Demo (
        )

system LgSvr1 (
        Capacity = 200
        Limits = { ShrMemSeg=15, Semaphores=30, Processors=18}
        LoadWarningLevel = 80
        LoadTimeThreshold = 900
        )

system LgSvr2 (
        Capacity = 200
        Limits = { ShrMemSeg=15, Semaphores=30, Processors=18 }
        LoadWarningLevel=80
        LoadTimeThreshold=900
        )

system LgSvr3 (
        Capacity = 200
        Limits = { ShrMemSeg=15, Semaphores=30, Processors=18 }
        LoadWarningLevel=80
        LoadTimeThreshold=900
        )

system MedSvr1 (
        Capacity = 100
        Limits = { ShrMemSeg=5, Semaphores=10, Processors=6}
        )

system MedSvr2 (
        Capacity = 100
```

```
        Limits = { ShrMemSeg=5, Semaphores=10, Processors=6 }
        )

    system MedSvr3 (
        Capacity = 100
        Limits = { ShrMemSeg=5, Semaphores=10, Processors=6 }
        )

    system MedSvr4 (
        Capacity = 100
        Limits = { ShrMemSeg=5, Semaphores=10, Processors=6 }
        )

    system MedSvr5 (
        Capacity = 100
        Limits = { ShrMemSeg=5, Semaphores=10, Processors=6 }
        )

    group Database1 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                       MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                       MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { LgSvr1, LgSvr2, LgSvr3 }
        FailOverPolicy = Load
        Load = 100
        Prerequisites = { ShrMemSeg=5, Semaphores=10, Processors=6 }
}
        )

    group Database2 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                       MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                       MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { LgSvr1, LgSvr2, LgSvr3 }
        FailOverPolicy = Load
        Load = 100
        Prerequisites = { ShrMemSeg=5, Semaphores=10, Processors=6 }
}
        )

    group Database3 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                       MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                       MedSvr3=1, MedSvr4=1, MedSvr5=1 }
```

```
        AutoStartPolicy = Load
        AutoStartList = { LgSvr1, LgSvr2, LgSvr3 }
        FailOverPolicy = Load
        Load = 100
        Prerequisites = { ShrMemSeg=5, Semaphores=10, Processors=6 }
}
                )

group Application1 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                        MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                        MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 }
        FailOverPolicy = Load
        Load = 50
                )

group Application2 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                        MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                        MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 }
        FailOverPolicy = Load
        Load = 50
                )


group Application3 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                        MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                        MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 }
        FailOverPolicy = Load
        Load = 50
                )

group Application4 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                        MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                        MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 }
        FailOverPolicy = Load
```

```
        Load = 50
                )

    group Application5 (
        SystemList = { LgSvr1, LgSvr2, LgSvr3, MedSvr1, MedSvr2, MedSvr3,
                        MedSvr4, MedSvr5 }
        SystemZones = { LgSvr1=0, LgSvr2=0, LgSvr3=0, MedSvr1=1, MedSvr2=1,
                        MedSvr3=1, MedSvr4=1, MedSvr5=1 }
        AutoStartPolicy = Load
        AutoStartList = { MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 }
        FailOverPolicy = Load
        Load = 50
                )
```

AutoStart Operation

Using the example configuration file above, the following AutoStart Sequence is possible:

Database1 – LgSvr1
Database2 – LgSvr2
Database3 – LgSvr3
Application1 – MedSvr1
Application2 – MedSvr2
Application3 – MedSvr3
Application4 – MedSvr4
Application5 – MedSvr5

Normal Operation

Assuming the above configuration, the following can be determined:

LgSvr1
AvailableCapacity=100
CurrentLimits = { ShrMemSeg=10, Semaphores=20, Processors=12 }

LgSvr2
AvailableCapacity=100
CurrentLimits = { ShrMemSeg=10, Semaphores=20, Processors=12 }

LgSvr3
AvailableCapacity=100
CurrentLimits = { ShrMemSeg=10, Semaphores=20, Processors=12 }

MedSvr1
AvailableCapacity=50
CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

MedSvr2
AvailableCapacity=50

CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

MedSvr3
AvailableCapacity=50
CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

MedSvr4
AvailableCapacity=50
CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

MedSvr5
AvailableCapacity=50
CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

Failure Scenario

The configuration above shows FailOverPolicy=Load and SystemZones. The database zone (System Zone 0) is capable of handling up to two failures. Each server has adequate Limits to support up to three database application groups (with an expected performance drop when all database application groups are running on one server). Similarly, the application zone has excess capacity built into each system.

In this example, each of MedSvr1 through MedSvr5 specifies Limits to support one database, even though the application groups G4 through G8 do not specify Prerequisites. This configuration allows a database to fail across SystemZones if absolutely necessary and run on the least loaded application zone machine.

For the first failure example, assume system LgSvr3 fails. The cluster management application scans all available systems in Database2's SystemList, with the same SystemZones grouping as LgSvr3. The cluster management application then creates a subset of systems meeting the application group's Prerequisites. In this case, LgSvr1 and LgSvr2 meet all necessary Limits, and Database1 is brought online on LgSvr1. The following configuration for the database zone is produced:

LgSvr1
AvailableCapacity=0
CurrentLimits = { ShrMemSeg=5, Semaphores=10, Processors=6 }

LgSvr2
AvailableCapacity=100
CurrentLimits = { ShrMemSeg=10, Semaphores=15, Processors=12 }

In this scenario, a further failure of a database can be tolerated, as each system has remaining Limits sufficient to accommodate the database application group running on the peer system.

Cascading Failure Scenario

If the performance of a specific database is unacceptable with two database groups running on one server (or three following a second failure), the SystemZones policy has another helpful effect. Failing a database group into the application zone has the effect of resetting its preferred zone. For example, in the above scenario, Database1 has been moved to LgSvr1. The administrator could reconfigure the application zone to move two application groups to one system. Then the database application can be switched to the empty application server (MedSvr1-MedSvr5). This will place Database1 in Zone1 (the application zone). If a failure occurs in Database1, the least-loaded server in the Application zone meeting its Prerequisites is selected as the failover target.